A highest-label preflow algorithm for the minimum flow problem

LAURA CIUPALĂ
University Transilvania of Braşov
Department of Computer Science
Iuliu Maniu Street 50, Braşov
ROMANIA

ELEONOR CIUREA
University Transilvania of Braşov
Department of Computer Science
Iuliu Maniu Street 50, Braşov
ROMANIA

Abstract: In this paper, we describe the highest-label preflow algorithm for minimum flow. This algorithm is a special implementation of the generic preflow algorithm developed by Ciurea and Ciupală in [7]. It examines always an active node with the highest distance label and runs in $O(n^2\sqrt{m})$ time.

Key–Words: Network flow; Network algorithms; Minimum flow problem

1 Introduction

The literature on network flow problem is extensive. Over the past 50 years researchers have made continuous improvements to algorithms for solving several classes of problems. From the late 1940s through the 1950s, researchers designed many of the fundamental algorithms for network flow, including methods for maximum flow and minimum cost flow problems. In the next decades, there are many research contributions concerning improving the computational complexity of network flow algorithms by using enhanced data structures, techniques of scaling the problem data etc.

Although it has its own applications, the minimum flow problem was not dealt so often as the maximum flow and the minimum cost flow problem. There are many problems that occur in economy that can be reduced to minimum flow problems.

The minimum flow problem in a network can be solved in two phases:
(1) establishing a feasible flow, if there is one
(2) from a given feasible flow, establish the minimum flow.

The problem of determining a feasible flow can be reduced to a maximum flow problem (for details see [1]).

For the second phase of the minimum flow problem there are three approaches:

1. using decreasing path algorithms (see [7], [8])
2. using preflow algorithms (see [7], [8])
3. finding a maximum flow from the sink node to the source node in the residual network (see [2], [5]).

The preflow algorithms for the minimum flow are more efficient than the decreasing path algorithms. In [7], Ciurea and Ciupală presented a generic preflow algorithm that runs in $O(n^2m)$ time and a special implementation of it: FIFO preflow algorithm that runs in $O(n^3)$.

In this paper, we develop a highest-label preflow algorithm for the minimum flow problem. This algorithm is a special implementation of the generic preflow algorithm for minimum flow (described in [7], [8]). As its name suggests, it always pulls flow from the active node with the highest distance label and it runs in $O(n^2\sqrt{m})$ time. Consequently, the highest-label preflow algorithm is faster than the FIFO preflow algorithm.

2 Notation and definition

We consider a capacitated network $G = (N, A, l, c, s, t)$ with a nonnegative capacity $c(i,j)$ and with a nonnegative lower bound $l(i,j)$ associated with each arc $(i,j) \in A$. We distinguish two special nodes in the network G : a source node s and a sink node t.

Let $\bar{c} = \max\{c(i,j) \mid (i,j) \in A\}$.

A *flow* is a function $f : A \to \Re_+$ satisfying the next conditions:

$$f(s,N) - f(N,s) = v \tag{1}$$

$$f(i,N) - f(N,i) = 0, i \neq s,t \tag{2}$$

$$f(t,N) - f(N,t) = -v \tag{3}$$

$$l(i,j) \leq f(i,j) \leq c(i,j), \forall (i,j) \in A \tag{4}$$

for some $v \geq 0$, where

$$f(i,N) = \sum_{j \mid (i,j) \in A} f(i,j), \forall i \in N$$

and

$$f(N,i) = \sum_{j \mid (j,i) \in A} f(j,i), \forall i \in N.$$

We refer to v as the *value* of the flow f.

The minimum flow problem is to determine a flow f for which v is minimized.

For the minimum flow problem, a *preflow* is a function $f : A \rightarrow \Re_+$ satisfying the next conditions:

$$f(i,N) - f(N,i) \leq 0, i \neq s, t \tag{5}$$

$$l(i,j) \leq f(i,j) \leq c(i,j), \forall (i,j) \in A \tag{6}$$

Let f be a preflow. We define the *deficit* of a node $i \in N$ in the following manner:

$$e(i) = f(i,N) - f(N,i) \tag{7}$$

Thus, for the minimum flow problem, for any preflow f, we have $e(i) \leq 0,\ i \in N \setminus \{s,t\}$.

We say that a node $i \in N \setminus \{s,t\}$ is *active* if $e(i) < 0$ and *balanced* if $e(i) = 0$.

A preflow f for which

$$e(i) = 0,\ i \in N \setminus \{s,t\}$$

is a flow. Consequently, a flow is a particular case of preflow.

For the minimum flow problem, the *residual capacity* $r(i,j)$ of any arc $(i,j) \in A$, with respect to a given preflow f, is given by

$$r(i,j) = c(j,i) - f(j,i) + f(i,j) - l(i,j).$$

By convention, if $(j,i) \notin A$ then we add arc (j,i) to the set of arcs A and we set $l\,(j,i) = 0$ and $c\,(j,i) = 0$. The residual capacity of the arc (i,j) represents the maximum amount of flow from the node i to node j that can be cancelled. The network $G_f = (N, A_f)$ consisting only of the arcs with positive residual capacity is referred to as the *residual network* (with respect to preflow f).

In the residual network G_f, the *distance function* $d : N \rightarrow \mathbf{N}$ with respect to a given preflow f is a function from the set of nodes to the nonnegative integers. We say that a distance function is *valid* if it satisfies the following conditions:

$$d(s) = 0$$

$$d(j) \leq d(i) + 1, \text{ for every arc}(i,j) \in A_f.$$

We refer to $d(i)$ as the distance label of node i.

Lemma 1 *[3](a) If the distance labels are valid, the distance label $d(i)$ is a lower bound on the length of the shortest directed path from node s to node i in the residual network.*

(b) If $d(t) \geq n$, the residual network contains no directed path from the source node to the sink node.

We say that the distance labels are *exact* if for each node i, $d(i)$ equals the length of the shortest path from node s to node i in the residual network.

We refer to an arc (i,j) from the residual network as an *admissible arc* if $d(j) = d(i) + 1$; otherwise it is *inadmissible*.

We refer to a node i with $e(i) < 0$ as an *active* node. We adopt the convention that the source node and the sink node are never active.

3 Highest-label preflow algorithm

In an iteration, the generic preflow algorithm for minimum flow (described in[7]) selects a node, say node i, and performs a canceling or a noncanceling pull, or relabels the node. If the algorithm performs a canceling pull, then node i might still be active, but, in the next iteration, the algorithm may select another active node for performing a pull or a relabel operation. We can establish the rule that whenever the algorithm selects an active node, it keeps pulling flow from that node until either its deficit becomes zero or the algorithm relabels the node. We refer to a sequence of canceling pulls followed either by a noncanceling pull or a relabel operation as a *node examination.*

The highest-label preflow algorithm for minimum flow, described in this paper, examines always an active node with the highest distance label. The algorithm maintains the set L of the active nodes as a priority queue, with priority d. It selects an active node j with the highest priority from the priority queue L, performs pulls from this node and adds newly active nodes to L. The algorithm terminates when the priority queue of active nodes is empty.

The highest-label preflow algorithm always pulls flow from an active node with the highest distance label. Let $h = \max\{d(i) \mid i \text{ is active}\}$. The algorithm first examines nodes with distance label equal to h and pulls the flow from these nodes to nodes with distance labels equal to $h - 1$ and, from these nodes, to the nodes with distance labels equal to $h - 2$ and so on until the algorithm relabels a node or it has analyzed all the active nodes. When it has relabeled a node, the algorithm repeats the same process. If the algorithm does not relabel any node during n consecutive node examinations, all the deficit reaches the source or the sink and the algorithm terminates.

The highest-label preflow algorithm for the minimum flow problem is the following:

```
**Highest-Label Preflow Algorithm;**
**Begin**
 let $f$ be a feasible flow in network $G$;
 compute the exact distance labels $d(\cdot)$ in the residual
 network $G_f$;
 **if** $t$ is not labeled
     **then** $f$ is a minimum flow
     **else begin**
         $L := \emptyset$;
         **for** each arc $(i,t) \in A$ **do**
         **begin**
             $f(i,t) := l(i,t)$;
             **if** $(e(i) < 0)$ **and** $(i \neq s)$
             **then** add $i$ with priority $d(i)$ to the priority
             queue $L$;
         **end;**
         $d(t) := n$;
         **while** $L \neq \emptyset$ **do**
         **begin**
             remove the node $j$ with the highest priority
             from the priority queue $L$;
             *pull/relabel(j);*
         **end**
     **end**
**end.**
```

```
**procedure** *pull/relabel(j)*;
**begin**
 select the first arc $(i,j)$ that enters in node $j$;
 $B := 1$;
 **repeat**
     **if** $(i,j)$ is an admissible arc
     **then begin**
             pull $g = \min(-e(j), r(i,j))$ units of flow
             from node $j$ to node $i$;
             **if** $(i \notin L)$ **and** $(i \neq s)$ **and** $(i \neq t)$
             **then** add $i$ with priority $d(i)$ to the priority
             queue $L$;
     **end;**
     **if** $e(j) < 0$
     **then if** $(i,j)$ is not the last arc that enters in $j$
         **then** select the next arc $(i,j)$ that enters in $j$
         **else begin**
                 $d(j) = \min\{d(i) \mid (i,j) \in A_f\} + 1$;
                 B:=0;
         **end;**
 **until** $(e(j) = 0)$ **or** $(B = 0)$;
 **if** $e(j) < 0$
     **then** add $j$ with priority $d(j)$ to the priority queue
     $L$;
 **end;**
```

Theorem 2 *The highest-label preflow algorithm computes correctly a minimum flow.*

Proof: The correctness of the highest-label preflow algorithm follows from the correctness of the generic preflow algorithm (see [7]). □

Actually, the algorithm terminates with optimal residual capacities. From these residual capacities we can determine a minimum flow in several ways. For example, we can make a variable change: For all arcs (i,j), let $c'(i,j) = c(i,j) - l(i,j)$, $r'(i,j) = r(i,j)$ and $f'(i,j) = f(i,j) - l(i,j)$. The residual capacity of arc (i,j) is

$$r(i,j) = c(j,i) - f(j,i) + f(i,j) - l(i,j).$$

Equivalently,

$$r'(i,j) = c'(j,i) - f'(j,i) + f'(i,j).$$

We can compute the value of f' in the following way:

$$f'(i,j) = \max(r'(i,j) - c'(j,i), 0)$$

Converting back into the original variables, we obtain the following expression:

$$f(i,j) = l(i,j) + \max(r(i,j) - c(j,i) + l(j,i), 0).$$

Theorem 3 *The highest-label preflow algorithm runs in $O(n^2\sqrt{m})$ time.*

Proof: This theorem can be proved in a manner similar to the proof of the complexity of the highest-label preflow algorithm for the maximum flow. (see [1]) □

4 Conclusion

In this paper, we described the highest-label preflow algorithm for minimum flow, that is a special implementation of the generic preflow algorithm developed by Ciurea and Ciupală in [7]. It examines always an active node with the highest distance label and it runs in $O(n^2\sqrt{m})$ time.

References:

[1] R. Ahuja, T. Magnanti and J. Orlin, *Network flows. Theory, algorithms and applications*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1993.

[2] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer-Verlag, London, 2001.

[3] L. Ciupală, A deficit scaling algorithm for the minimum flow problem, *Sadhana* 31, Part 3, 2006, pp.1169-1174.

[4] L. Ciupală, A scaling out-of-kilter algorithm for minimum cost flow, *Control and Cybernetics* 34, no.4, 2005, pp. 1169-1174.

[5] L. Ciupală and E. Ciurea, An algorithm for the minimum flow problem, *The Sixth International Conference of Economic Informatics*, 2003, pp. 565-569.

[6] L. Ciupală and E. Ciurea, An approach of the minimum flow problem, *The Fifth International Symposium of Economic Informatics*, 2001, pp. 786-790.

[7] E. Ciurea and L. Ciupală, Sequential and parallel algorithms for minimum flows, *Journal of Applied Mathematics and Computing* 15, no.1-2, 2004, pp. 53-78.

[8] E. Ciurea and L. Ciupală, Algorithms for minimum flows, *Computer Science Journal of Moldova* 9, no.3(27), 2001, pp. 275-290.

[9] A. V. Goldberg and R. E. Tarjan, *A New Approach to the Maximum Flow Problem*, Journal of ACM 35, 1988, pp. 921-940.